Assignments 7 and 8:

Q1. a) Undefined, a) 8, b) 8, c) 9, d) 10, e) 1

Q2. Global scope contains variables which are declared outside a function, and can be accessed both inside and outside the function, whereas local scope constitutes variables declared within the function, and are accessible by the function, or the block.

Q3. a) No, b) Yes, c) No, d) Yes, e) Yes

Q4. a) 81, b) 25

Q5. 10

Q6.   
// Revealing Module Pattern

var count = function() {

var counter = 0;

const getCounter = function() {

return counter;

}

const add = function() {

return counter += 1;

}

const reset = function() {

counter = 0;

}

return {

getCounter,

add,

reset

}

} ();

Q7. The counter variable inside the function is a free variable, and even though it is defined outside the block of execution, it can be accessed by the instances within the block by reference even after execution.

Q8.  
make\_adder : function(inc) {

return counter + inc;

}

Q9. To remove all the added names and from the global namespace, the code can be wrapped in a “module pattern”.

Q10.  
var Employee = function() {

var name;

var age;

var salary;

function getName(){

return this.name;

}

function getAge() {

return this.age;

}

function getSalary() {

return this.salary;

}

function publicSetName(newName){

this.Name = newName;

}

function publicSetAge(newAge) {

this.age = newAge;

}

function publicsetSalary(newSalary) {

this.salary = newSalary;

}

function publicIncreaseAge() {

this.age += 1;

function publicIncreaseSalary(percentage) {

this.salary = getSalary() \* (1 + percentage/100);

}

return {

setName: publicSetName;

setAge: publicSetAge,

setSalary: publicSetSalary,

increaseAge: publicIncreaseAge,

increaseSalary: publicIncreaseSalary;

};

}();

Q11. Employee.prototype.address = “default”

Employee.prototype.setAddress(newAddress) {

this.address = newAddress;

}

Employee.prototype.getAddress() {

return this.address;

}